**目录**

[多路数据采集控制系统 1](#_Toc24498475)

[1 设计题目 1](#_Toc24498476)

[2 设计目的 1](#_Toc24498477)

[3 设计内容及要求 1](#_Toc24498478)

[4 设计过程 1](#_Toc24498479)

[4.1 总体设计 1](#_Toc24498480)

[4.2 关键元件选取 2](#_Toc24498481)

[4.3 具体电路设计，基于Proteus8 6](#_Toc24498482)

[4.4 程序设计 10](#_Toc24498483)

[4.5 主要程序介绍 12](#_Toc24498484)

[5 仿真过程，Proteus 8与uVersion联调 16](#_Toc24498485)

[6 遇到的问题与解决方法 22](#_Toc24498486)

[7 心得体会 22](#_Toc24498487)

[附件 24](#_Toc24498488)

**图目录**

[图 1-系统框图 1](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498489)

[图 2- STC89C52引脚图 2](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498490)

[图 3- STC89C52实物图 2](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498491)

[图 4- STC89C52内部结构 2](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498492)

[图 5-ADC0809实物图 3](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498493)

[图 6-ADC0809引脚 3](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498494)

[图 7-对应引脚 4](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498495)

[图 8-背部视图 4](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498496)

[图 9-LCD俯视图 4](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498497)

[图 10-LCD实物图 4](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498498)

[图 11-74HC238真值表 5](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498499)

[图 12-实物图 5](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498500)

[图 13-引脚图 5](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498501)

[图 14-晶振模块 6](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498502)

[图 15-复位模块 6](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498503)

[图 16-ADC模块 7](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498504)

[图 17-LCD显示 7](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498505)

[图 18-译码器与LED报警 8](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498506)

[图 19-按键模块 9](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498507)

[图 20-定时器T0中断 10](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498508)

[图 21-外部中断T1 10](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498509)

[图 22-外部中断T0 10](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498510)

[图 23-主程序流程图 11](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498511)

[图 24-写操作时序图 13](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498512)

[图 25-激活uVersion4中的Proteus选项 16](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498513)

[图 26-启用Enable Remote 16](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498514)

[图 27-全速运行 17](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498515)

[图 28-点击调试 17](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498516)

[图 29-编译通过 17](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498517)

[图 30-点击编译 17](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498518)

[图 31-全速运行 18](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498519)

[图 32-输入8个不同的电压值 18](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498520)

[图 33-LED报警电路 19](file:///C:\Users\Jiang%20Hanlin\Desktop\多通道采集控制系统.docx#_Toc24498521)

**摘要**

随着经济的飞速发展和科学技术水平的不断提高,智能数据采集系统在工业生产以及科学研究中得到了广泛的应用.在信息化时代,数据和信息无疑成为一种重要的资源,而数据采集系统的出现更是进一步促进了人机交互、对设备的自动检测控制等的实现,为现代化工业生产提供了方便。本次课程设计将阐述基于单片机的智能数据采集系统的设计要点及其具体方法,以期对基于单片机的智能数据采集系统的改造和创新做出应有的贡献

此次课程设计利用型号为STC89C52RC的单片机，ADC0809，LCD1602等电子元器件来实现对8通道模拟电压的采集的功能。

**关键字**：数据采集系统、单片机、STC89C52、模数转换、lcd显示

**Abstract**

With the rapid development of economy and constantly improve the level of science and technology, intelligent data acquisition system in the industrial production and scientific research has been widely used. In the era of information, data and information has become a kind of important resources, and the emergence of the data acquisition system is further promoting the human-computer interaction, the realization of the automated testing of equipment control and so on, provides the convenience for modern industrial production. This course design will elaborate the design points and specific methods of the intelligent data acquisition system based on the single chip microcomputer, in order to make due contributions to the transformation and innovation of the intelligent data acquisition system based on the single chip microcomputer

This course design USES the single-chip microcomputer model STC89C52RC, ADC0809, LCD1602 and other electronic components to achieve the function of 8 channel analog voltage acquisition.

**Keywords**: Data acquisition system, single chip microcomputer, STC89C52, analog-to-digital conversion, LCD display

多路数据采集控制系统

# 设计题目

单片机多路数据采集控制系统—Proteus软件仿真

# 设计目的

运用单片机原理及其应用等课程知识,根据题目要求进行进行软件仿真的设计和调试,从而加深对本课程知识的理解, 把学过的比较零碎的知识系统化，比较系统的学习开发单片机应用系统的基本步骤和基本方法，使学生应用知识能力、设计能力、调试能力以及报告撰写能力等有一定的提高。

# 设计内容及要求

用8051单片机设计数据采集控制系统，基本要求如下：

1、 可实现8路数据的采集，假设８路信号均为0-5V的电压信号；

2、 采集数据可通过LCD显示，显示格式为：[通道号] 电压值，如[０１]： ４.5V

3、 可通过键盘设置采集方式；（单点采集、多路巡测、采集时间间隔\*）

4、 具有异常数据声音报警功能：对第一路数据可设置正常数据的上限值和下限值，当采集的数据出现异常，发出报警信号。

# 设计过程

## 总体设计

此次多路数据采集控制系统包括：

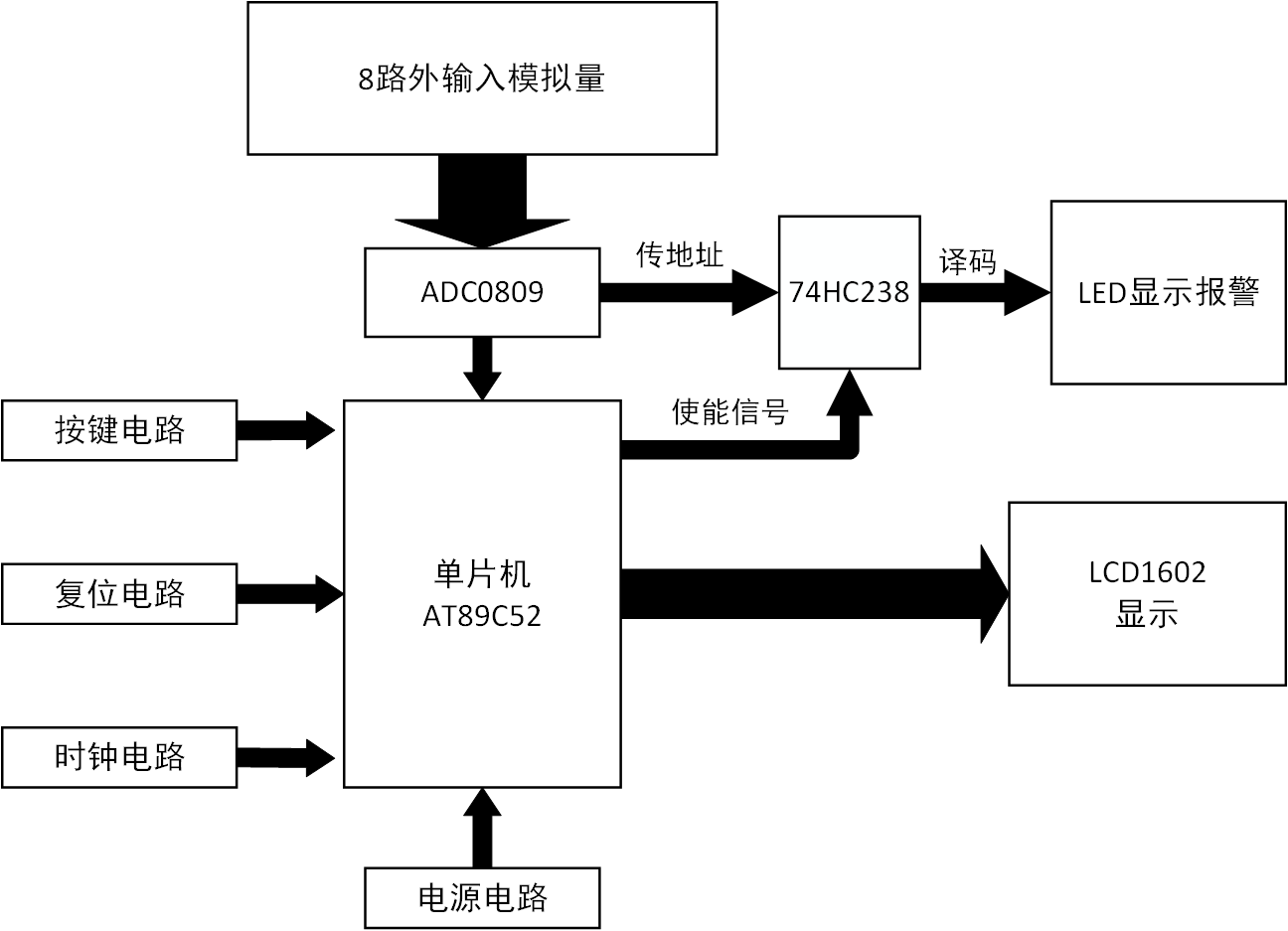
单片机为STC89C52RC，8位的AD转换芯片0809，LCD1602液晶显示屏，按键电路由按键、上拉电阻构成，晶振选择外接12MHz，部分电容，电阻，二极管。其电路总设计框图如下：

图 1-系统框图

## 关键元件选取

1. STC89C52RC

STC89C52RC是STC公司生产的一种低功耗、高性能CMOS8位微控制器，具有8K字节系统可编程Flash存储器。STC89C52使用经典的MCS-51内核，但是做了很多的改进使得芯片具有传统的51单片机不具备的功能。在单芯片上，拥有灵巧的8 位CPU 和在系统可编程Flash，使得STC89C52为众多嵌入式控制应用系统提供高灵活、超有效的解决方案。

32 位I/O 口线，看门狗定时器，内置4KB EEPROM，MAX810复位电路，3个16 位定时器/计数器，4个外部中断，一个7向量4级中断结构（兼容传统51的5向量2级中断结构），全双工串行口。另外 STC89C52 可降至0Hz 静态逻辑操作，支持2种软件可选择节电模式。空闲模式下，CPU 停止工作，允许RAM、定时器/计数器、串口、中断继续工作。

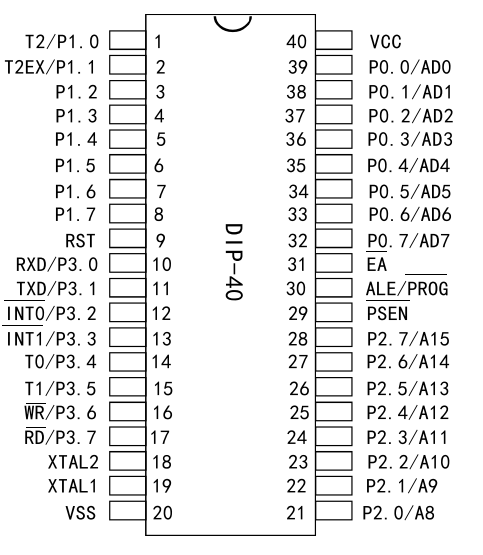


图 2- STC89C52引脚图



图 3- STC89C52实物图

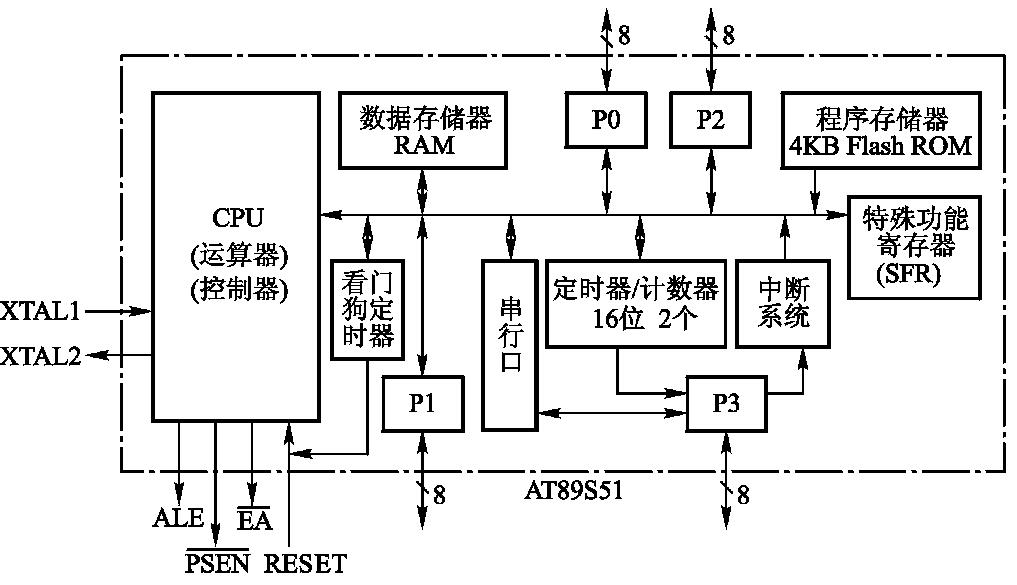


图 4- STC89C52内部结构

1、数据存储器（RAM）：片内为128B（52子系列为256B），片外最多可扩64KB。片内128B的RAM以高速RAM的形式集成，可加快单片机运行的速度和降低功耗

2、程序存储器（Flash ROM）：片内集成有8KB的Flash存储器，如片内容量不够，片外可外扩至64KB

3、中断系统：具有6个中断源，2级中断优先权

4、定时器/计数器：2个16位定时器/计数器（52子系列有3个），4种工作方式

5、看门狗定时器WDT：当CPU由于干扰使程序陷入死循环或跑飞时，WDT可使程序恢复正常运行

关键引脚

RESET：复位信号输入，在引脚加上持续时间大于2个机器周期的高电平，可使单片机复位。正常工作，此脚电平应 ≤ 0.5V。当看门狗定时器溢出输出时，该脚将输出长达96个时钟振荡周期的高电平

：1，读取内部存储器。0，读取外部存储器。

ALE：为CPU访问外部程序存储器或外部数据存储器提供地址锁存信号，将低8位地址锁存在片外的地址锁存器中。

1. ADC0809

ADC0809是采用COMS工艺制造的双列直插式单片8位A/D转换器。分辨率8位，精度7位，带8个模拟量输入通道，有通道地址译码锁存器，输出带三态数据锁存器。启动信号为脉冲启动方式，最大可调节误差为±1LSB。

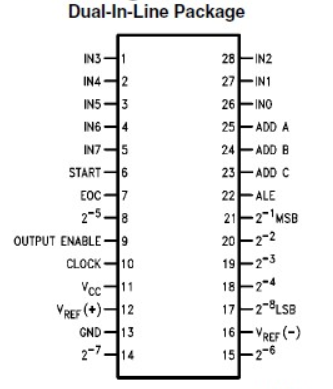
ADC0809内部没有时钟电路，故CLK时钟需由外部输入，fclk允许范围为500kHz-1MHz,典型值为640kHz。每通道的转换需要66~73个时钟脉冲，大约100-110us。（转换时间）工作温度范围为-40℃—+85℃。功耗为15mW，输入电压范围为0–5V，单一+5V电源供电。

图 5-ADC0809实物图

图 6-ADC0809引脚

IN0~IN7：8路模拟信号输入端。D0~D7：8位数字量输出端。START：启动控制输入端，高电平有效，用于启动ADC0809内部的A/D转换过程。ALE：地址锁存控制输入端，ALE端可与START端连接在一起，通过软件输入一个正脉冲，可立即启动A/D转换。EOC：转换结束信号输出端，开始A/D转换时为低电平，转换结束是输出高电平。OE：输出允许控制端，用于打开三态输出锁存器，当OE为高电平时，打开三态数据输出锁存器，将转换后的数据凉输送到数据总线上。CLK：始终信号输入端。ADDA(ADDB、ADDC)：8路模拟选通开关的3位地址选通输入端。

1. LCD1602

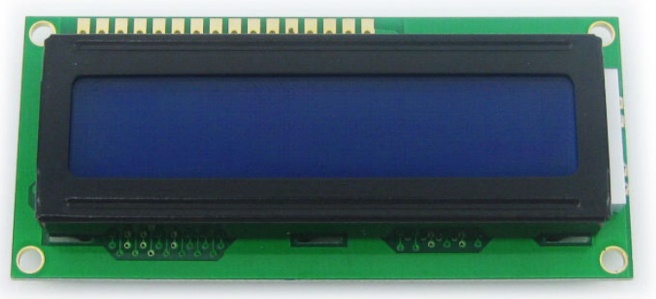
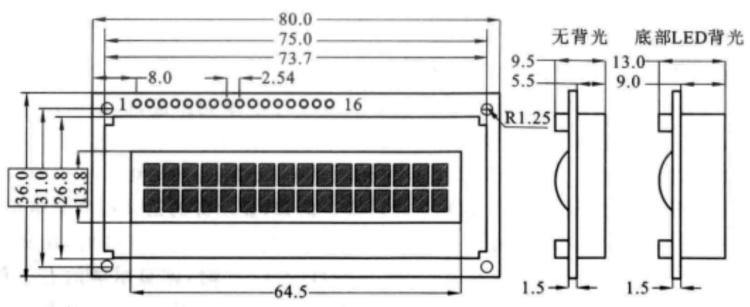
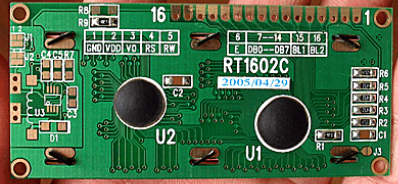
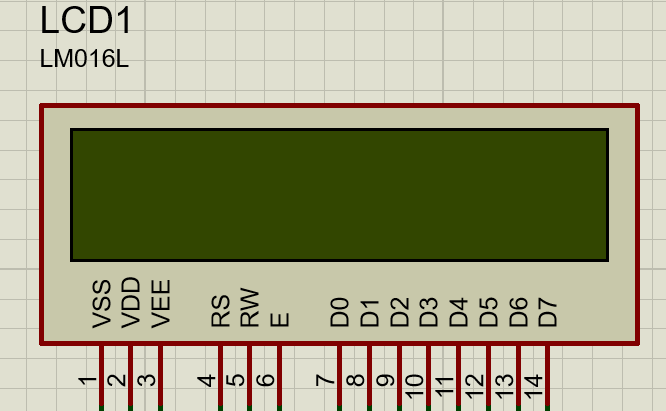
LCD1602液晶显示器是广泛使用的一种字符型液晶显示模块。它是由字符型液晶显示屏（LCD）、控制驱动主电路HD44780及其扩展驱动电路HD44100，以及少量电阻、电容元件和结构件等装配在PCB板上而组成。

图 7-对应引脚

图 8-背部视图

图 9-LCD俯视图

图 10-LCD实物图

·引脚1：VSS为地电源。

·引脚2：VDD接5V正电源。

·引脚3：VL为液晶显示器对比度调整端，接正电源时对比度最弱，接地时对比度最高，对比度过高时会产生“鬼影”现象，使用时可以通过一个10kQ的电位器调整其对比度。

·引脚4：RS为寄存器选择脚，高电平时选择数据寄存器、低电平时选择指令寄存器。

·引脚5：R/W为读/写信号线，高电平时进行读操作，低电平时进行写操作。当RS和R/W共同为低电平时可以写入指令或显示地址；当RS为低电平，R/W为高电平时，可以读忙信号；当RS为高电平，R/W为低电平时，可以写入数据。

·引脚6：E端为使能端，当E端由高电平跳变为低电平时，液晶模块执行命令。

·引脚7~14：D0~D7为8位双向数据线。

·引脚15：背光源正极。

·引脚16：背光源负极

1. 74HC238

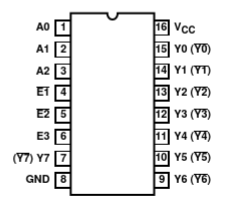
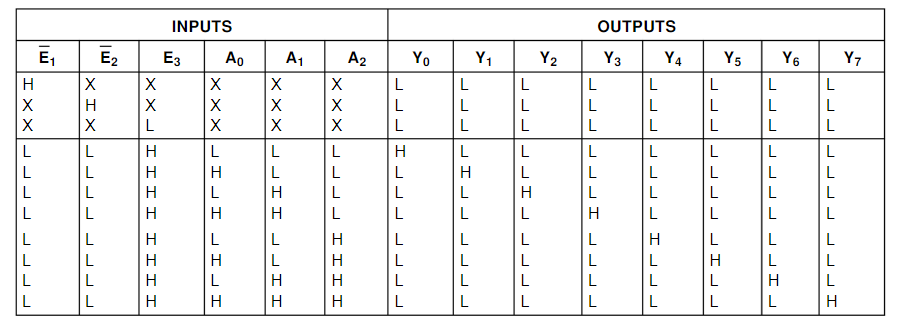
74HC238能够将3地址输入，译码出8个输出，输出的电平刚好和74LS138相反。

图 11-74HC238真值表

图 12-实物图

图 13-引脚图

用作IO扩展与复用

用3个IO，可以控制8个输出

A0~A2：3个输入

E1、E2：拉低使能，可以接地

E3：拉高使能，可以接VCC

Y0~Y7：8个输出

## 具体电路设计，基于Proteus8

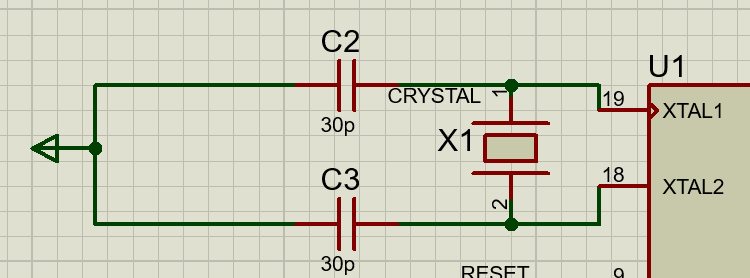
1. **晶振模块**

图 14-晶振模块

晶振与单片机的脚XTAL0和脚XTAL1构成的振荡电路中会产生谐波（也就是不希望存在的其他频率的波）这个波对电路的影响不大，但会降低电路的时钟振荡器的稳定性，为了电路的稳定性起见 ATMEL公司只是建议在晶振的两引脚处接入两个10pf-50pf的瓷片电容接地来削减谐波对电路的稳定性的影响，所以晶振所配的电容在10pf-50pf之间都可以的。

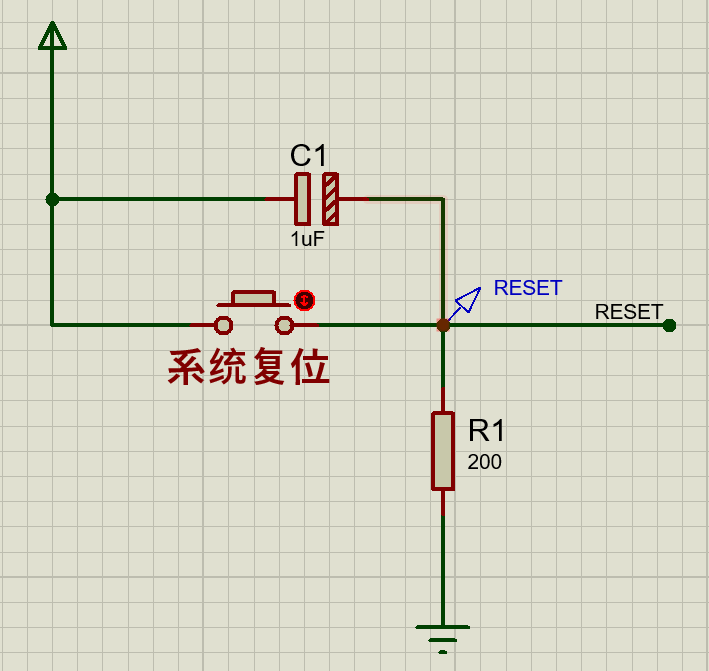
1. **单片机复位模块**

图 15-复位模块

复位电路工作原理如上所示，VCC上电时，使极性电容C充电，在200欧电阻上出现高电位电压，使得单片机复位；几个毫秒后，C充满，200欧电阻上电流降为0，电压也为0，使得单片机进入工作状态。工作期间，按下S1，C放电，放电结束后，又充电，在200电阻上出现高电压，使得单片机进入复位状态，直到S1松手，C又充电完毕，随后，单片机进入工作状态。

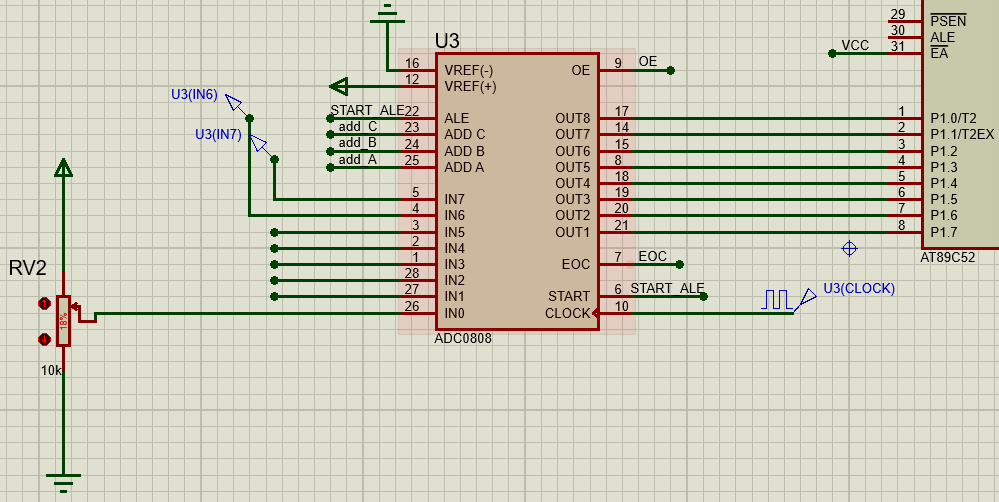
1. **ADC模块**

图 16-ADC模块

如上图，ADC0809的地址输入CBA、ALE、EOC、START、OE由AT89C52上的IO引脚来控制。8位OUT输出直接由AT89C52的P1口来接收

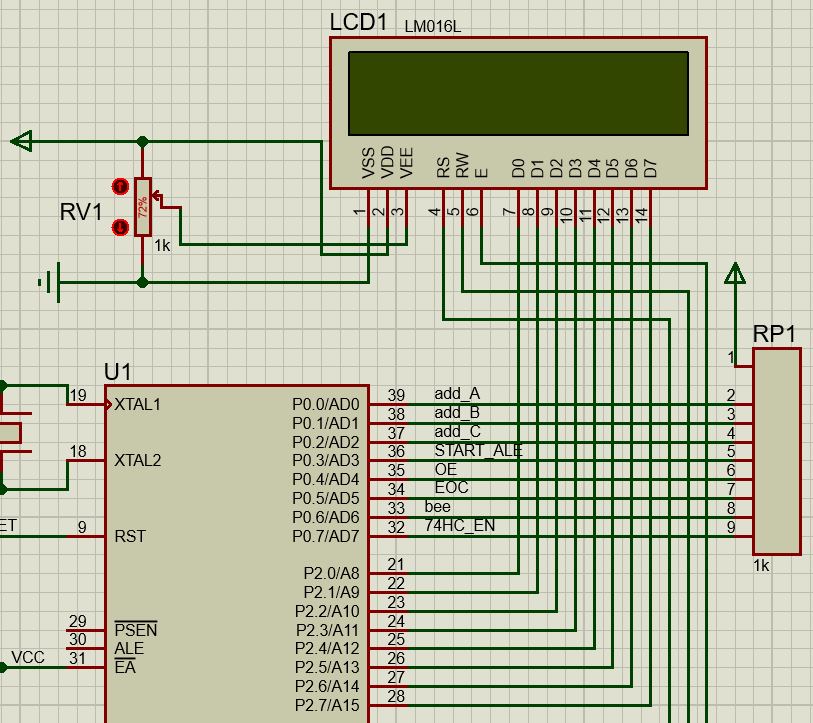
1. **LCD显示模块**

图 17-LCD显示

如上图所示Vss、Vdd、Vee构成分压，STC89C52RC的P2口驱动D0~D1，另外的3个IO分别控制RS、RW、E。

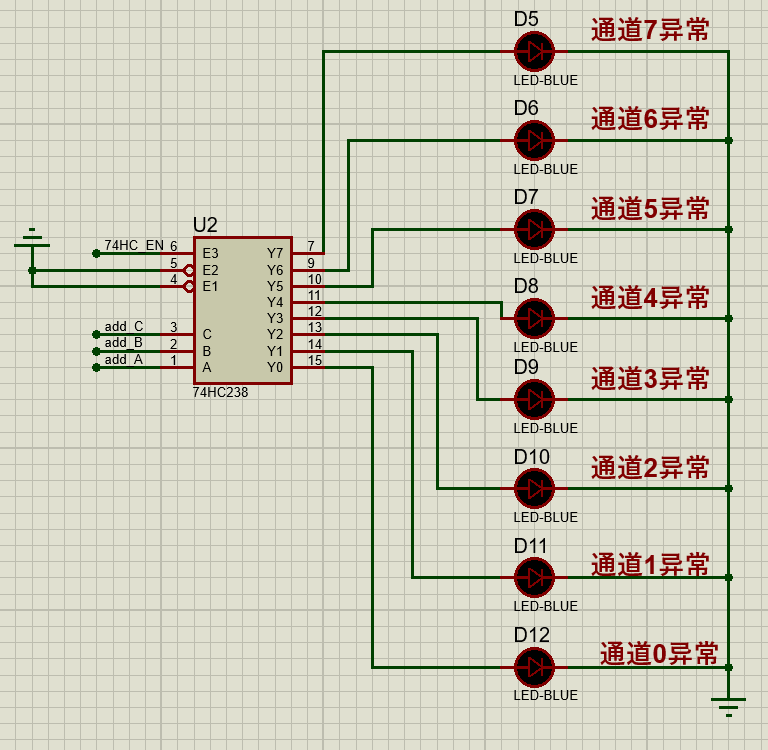
1. **译码器与LED报警模块**

图 18-译码器与LED报警

如上图所示，74HC238的CBA由ADC0809的CBA传输而来，使能端E3交由AT89C52的一个引脚控制，一旦切换到对应通道，如果该通道的电压异常，那么久点亮该通道的LED，进行报警。

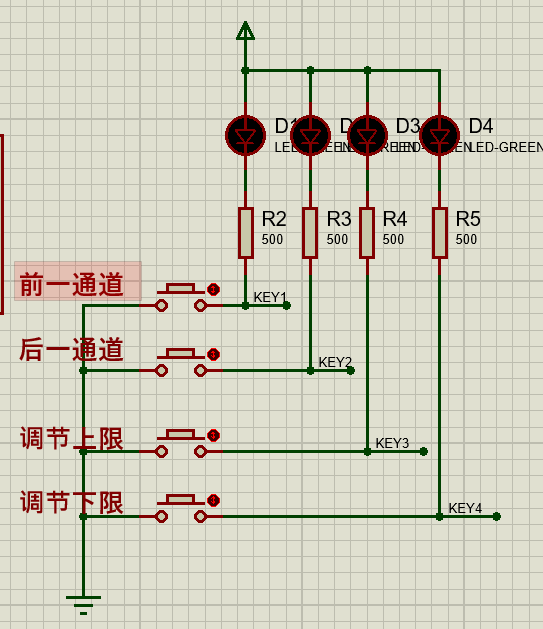
1. **按键模块**

图 19-按键模块

按键模块如上图所示，上拉电阻在上电时默认将按键右端置为高电平。一旦按键按下，电阻下方为低电平并传给STC89C52对应引脚，执行操作。

## 程序设计

此次的程序设计借助uVersion4实现。自定义了7个头文件，1个全局变量

它们分别为：

1、#include "AT89C52\_init.h" //AT89C52初始化相关实现

2、#include "lcd\_init.h" // LCD初始化相关实现

3、#include "lcd\_display.h" // LCD显示相关功能实现

4、#include "delay.h" //延时函数

5、#include "adjust\_voltage.h" //电压调节相关功能实现

6、#include "key\_check.h" //按键检测相关功能实现

7、#include "led\_warning.h" //led报警显示相关功能实现

8、unsigned char flag=0; //全局通道号

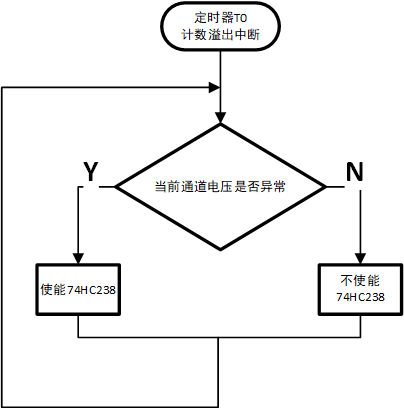
以下是本次程序设计的流程图，分为主函数执行流程图和中断函数执行流程图：

图 20-定时器T0中断

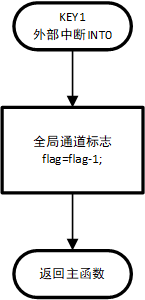
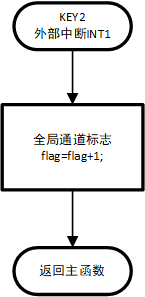


图 21-外部中断T1

图 22-外部中断T0

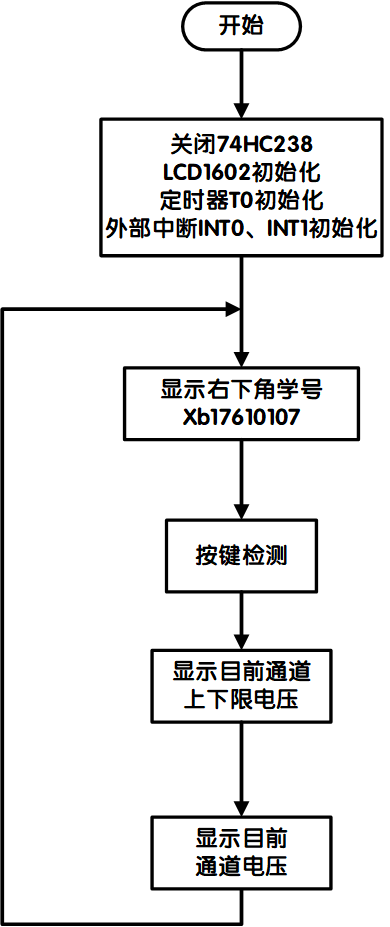


图 23-主程序流程图

## 主要程序介绍

1. **LCD显示程序介绍**

#include <reg52.h>

#include "lcd\_init.h"

#include "delay.h"

sbit LCD\_RS=P3^7;

sbit LCD\_RW=P3^6;

sbit LCD\_EN=P3^5;

sbit add\_A=P0^0;

sbit add\_B=P0^1;

sbit add\_C=P0^2;//ABC为3为地址输入线，译码出8位通道

sbit START\_ALE=P0^3;//地址锁存，锁存CBA地址

sbit OE=P0^4;//转换后数据输出使能

sbit EOC=P0^5;//是否转换完成，0正忙，1转换成功

void lcd\_init()

{

LCD\_Write\_Command(0x38);//功能设置指令6：8位数据，双行显示，5\*7字形

LCD\_Write\_Command(0x0c);//显示开关控制指令4：开启显示屏，关光标，光标不闪烁

LCD\_Write\_Command(0x06);//置输入模式指令3：数据读写后光标右移，画面不移动

LCD\_Write\_Command(0x01);//清屏指令1

}

void LCD\_Write\_Command(unsigned char com)//写命令函数

{

LCD\_RS=0;//选择指令寄存器

LCD\_RW=0;//选择写

P2=com;//传命令

delay(5);

LCD\_EN=1;//将使能线置高

delay(5);//将高电平延时1段时间

LCD\_EN=0;//产生下降沿，让LCD执行命令

}

void LCD\_Write\_Value(unsigned char value)//写数据函数

{

LCD\_RS=1;//选择数据寄存器

LCD\_RW=0;//选择写

P2=value;//传输显示数据

delay(5);

LCD\_EN=1;//将使能线置高

delay(5);//将高电平延时1段时间

LCD\_EN=0;//产生下降沿，让LCD执行命令

}

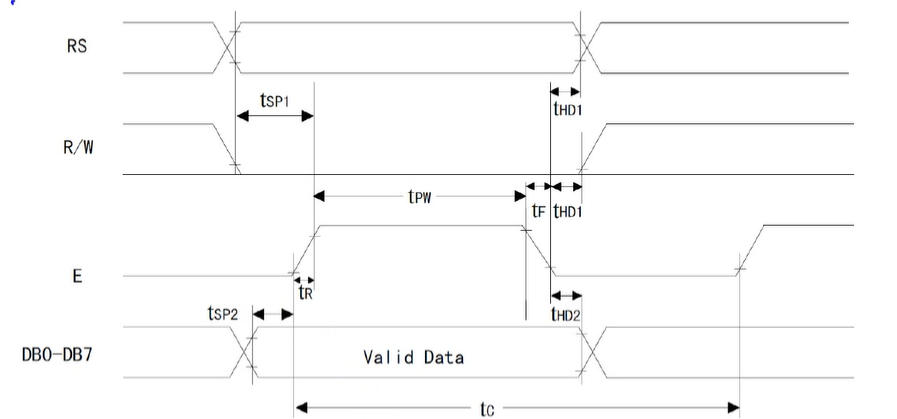
上述代码为LCD1602相关的所有命令。LCD1602可以显示2\*16个字符，每个字符又是5\*7的点阵组成。每个字符都有固定的显示段码，想要在LCD1602上指定位置显示指定字符分2个步骤，首先需要写命令到指令寄存器，然后再写数据到数据寄存器，才可以正确显示。以下是LCD1602写操作的时序图。

图 24-写操作时序图

RS为寄存器选择脚，1时选择数据寄存器、0时选择指令寄存器

R/W为读/写信号线，1时进行读操作，0时进行写操作。当RS和R/W共同为0时可以写入指令或显示地址；当RS=1，R/W=0时，可以写入数据。

如果是写命令或显示地址，刚开始E=0，RS=0，RW=0，之后P2将数据传给D0~D7，接下来E=1延时一段时间将数据读入，之后E=0命令或地址传输完成。

如果是写显示数据，刚开始E=0，RS=1，RW=0，之后P2将数据传给D0~D7，接下来E=1延时一段时间将数据读入，之后E=0数据传输完成。

D0~D7一共是8位，CGROM用来保存已有字符，写已有字符操作时，最高位D71，剩下的7位，若是写显示地址，其他7位为对应LCD点阵的地址外还要加上0x80H即D7；若是写数据，7位为对应CGROM中的字符的译码即可，查表可得。

CGRAM用来保存用户自定义的字符，但是只有8个空间。使用CGRAM时，首先得写CGRAM对应字符的地址，之后将自定义字符写入CGRAM。之后再写显示地址命令，然后写入显示的是CGRAM的8个字符中的第几个字符。

1. **ADC转化程序介绍**

START\_ALE=0;

OE=0;

select\_channel(1);//选择通道1进行转化

START\_ALE=1;

delay(5);

START\_ALE=0;

delay(5);//等待AD转换完成

while(EOC==1)

{

P1=0xff;//读数据之前将P1口的电压置高

OE=1;

temp1=P1/51.0;//将转换后的16进制准换成电压

temp2=temp1\*100;

delay(5);

OE=0;

break;

}

void select\_channel(unsigned char n)//通道选择函数

{

switch(n)

{

case 0: add\_C=0; add\_B=0; add\_A=0; break;//开启通道0

case 1: add\_C=0; add\_B=0; add\_A=1; break;//开启通道1

case 2: add\_C=0; add\_B=1; add\_A=0; break;//开启通道2

case 3: add\_C=0; add\_B=1; add\_A=1; break;//开启通道3

case 4: add\_C=1; add\_B=0; add\_A=0; break;//开启通道4

case 5: add\_C=1; add\_B=0; add\_A=1; break;//开启通道5

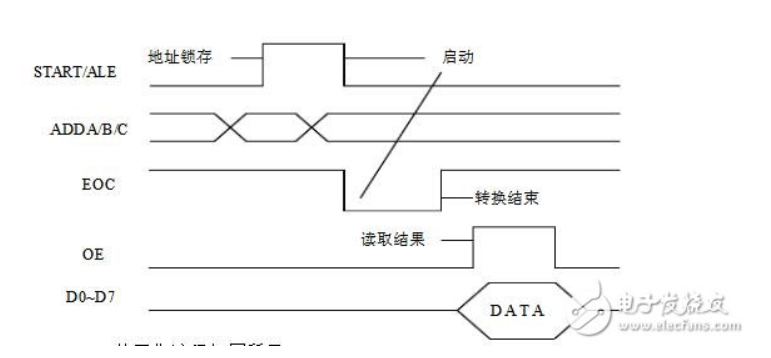
case 6: add\_C=1; add\_B=1; add\_A=0; break;//开启通道6

case 7: add\_C=1; add\_B=1; add\_A=1; break;//开启通道7

}

}

模数转换部分核心程序如上所示，ADC0809的START和ALE引脚由AT89C52一个引脚同时控制，unsigned float temp1用来保存转换以后的电压值，temp2用来保存扩大100倍后的temp1。add\_C、add\_B、add\_A为地址输入，以下是ADC0809的时序图。

模数转换的编程思想按照ADC0809的时序图来进行。

刚开始，OE=1，关闭三态输出锁存器。输入3位地址ABC，8种不同的组合对应不同的通道。将START=1，ALE=1，将地址存入地址锁存器中，经地址译码器译码从8路模拟通道中选通一路模拟量送到比较器，同时由于START上升沿使逐次逼近寄存器复位。

延时一段时间，START=0，下降沿启动A/D转换，转换期间EOC为低电平。

当转换结束时，转换的结果送入到输出三态锁存器，并使EOC信号自动回到高电平，通知CPU已转换结束。程序中判断EOC是否为1，如果EOC=1，将OE=1，单片机从输出端D0~D7读取数据，并延时一段时间后OE=0，准备下一次AD转换。

注意到在读取数据之前，将P2=0xff的原因是防止外部干扰。

# 仿真过程，Proteus 8与uVersion联调

图 25-激活uVersion4中的Proteus选项

图 26-启用Enable Remote

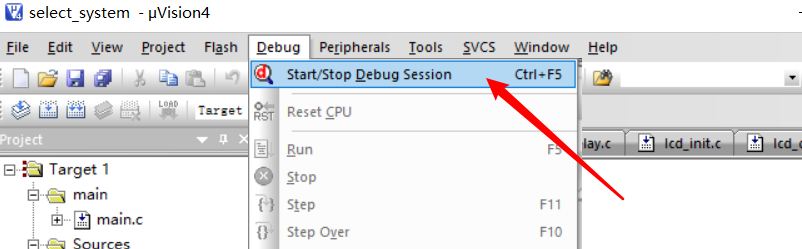
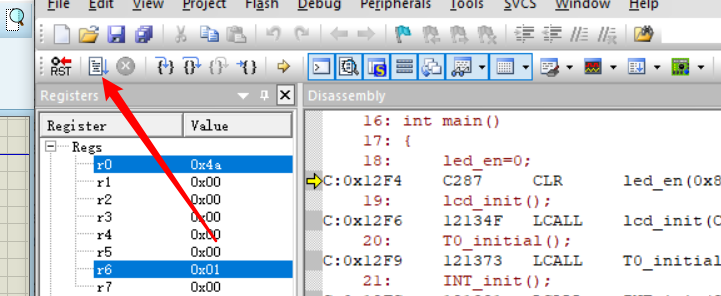
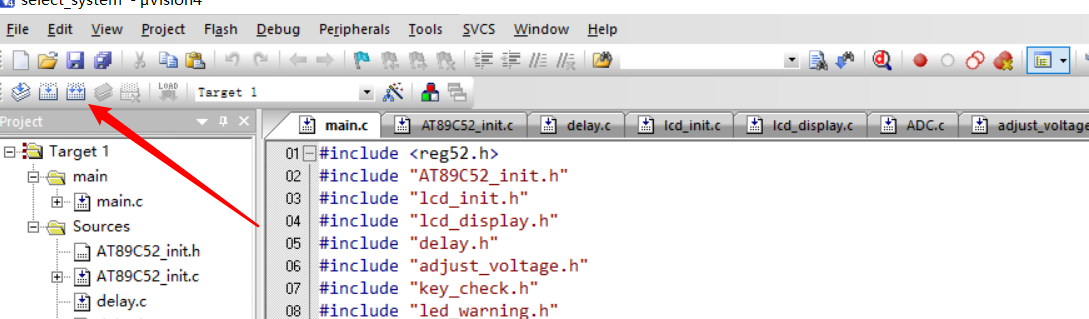
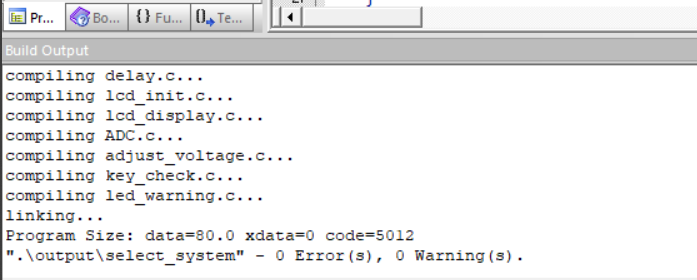
 

图 27-全速运行

图 28-点击调试

图 29-编译通过

图 30-点击编译

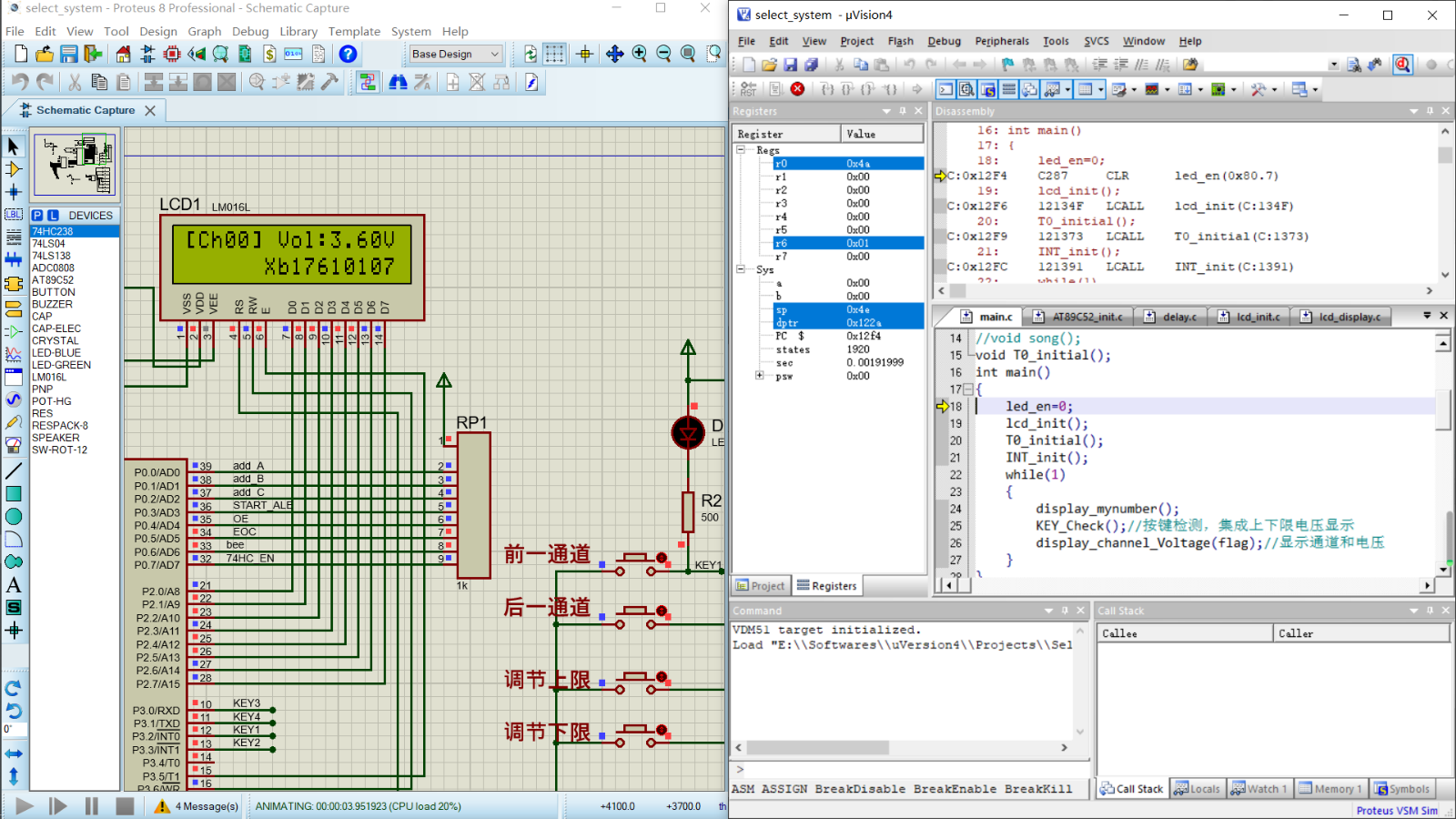


图 31-全速运行

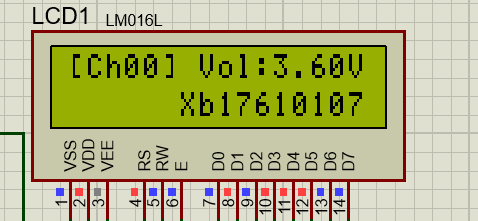
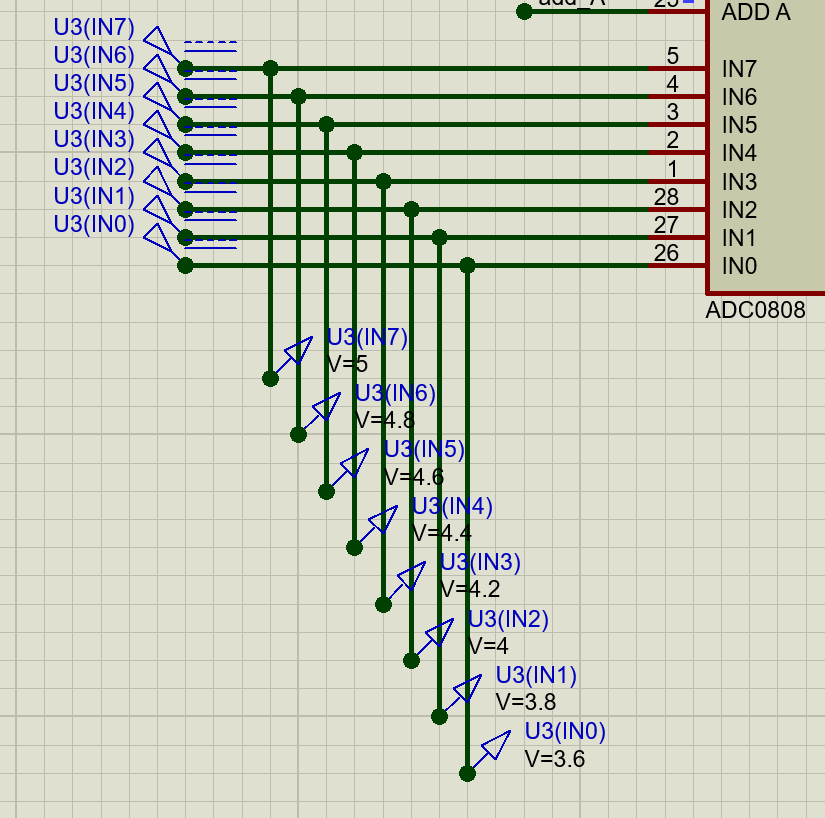
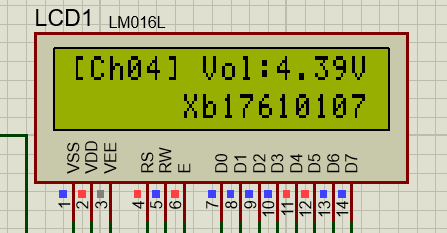
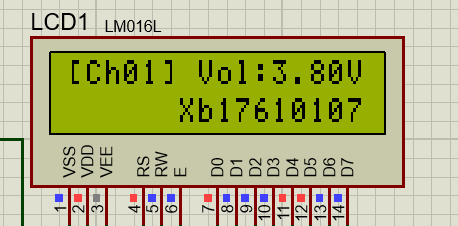
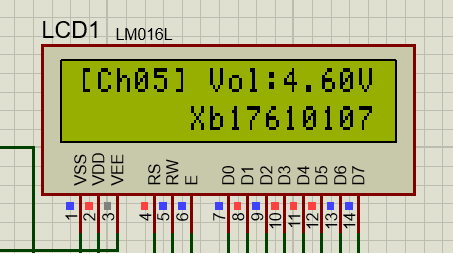
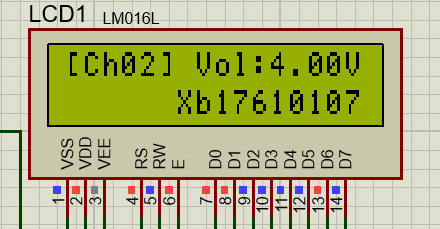
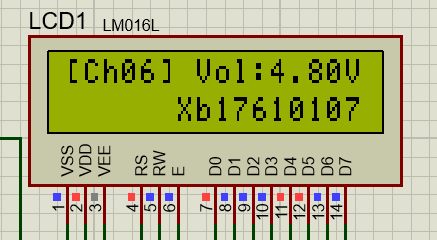
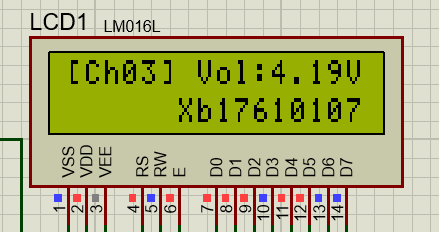
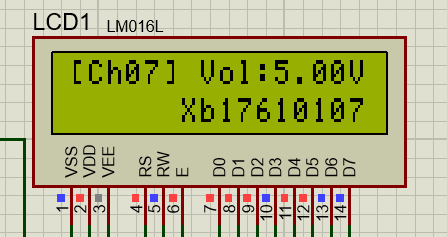


图 32-输入8个不同的电压值









如上图所示，IN0~IN7的电压为3.6~5，公差为0.2的等差数列，ADC0809工作正常，LCD能够显示对应通道的电压值。至于通道的切换由按键实现。

接下来是通道的上下限电压调节，我能够对分别对8个通道的上下限电压进行控制，实验要求利用蜂鸣器实现只对0通道的异常电压声音报警。在仿真的过程中，我发现蜂鸣器分为有源和无源2种，有源只要加上电压就会发出声音；而无源则需要PWM驱动。要实现发出音乐报警的功能对于毫无音乐天赋的我来讲太难了。于是我突发奇想，换了一种思维方式，利用LED进行显示报警，而且我不光做到了对单一通道的检测，还可以对任意当前通道的显示报警，实现该功能所需要的元器件是74HC238，它的输出刚好和74LS138相反，刚好符合检测特点。碰巧的是，ADC0809的3位地址可以与74HC238共用，接下来剩下三个使能端，2个接地，一个交由AT89C52的一个IO控制即可。仿真电路如下：

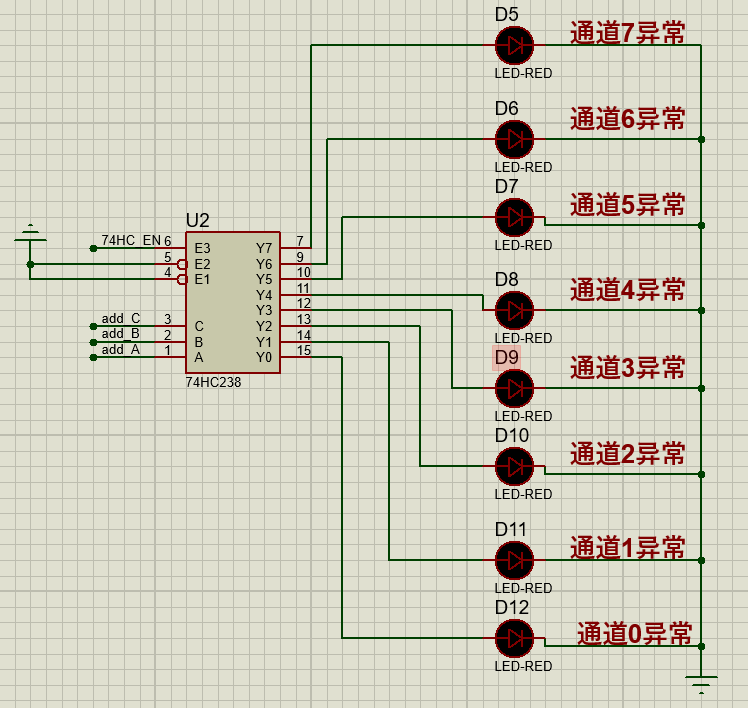
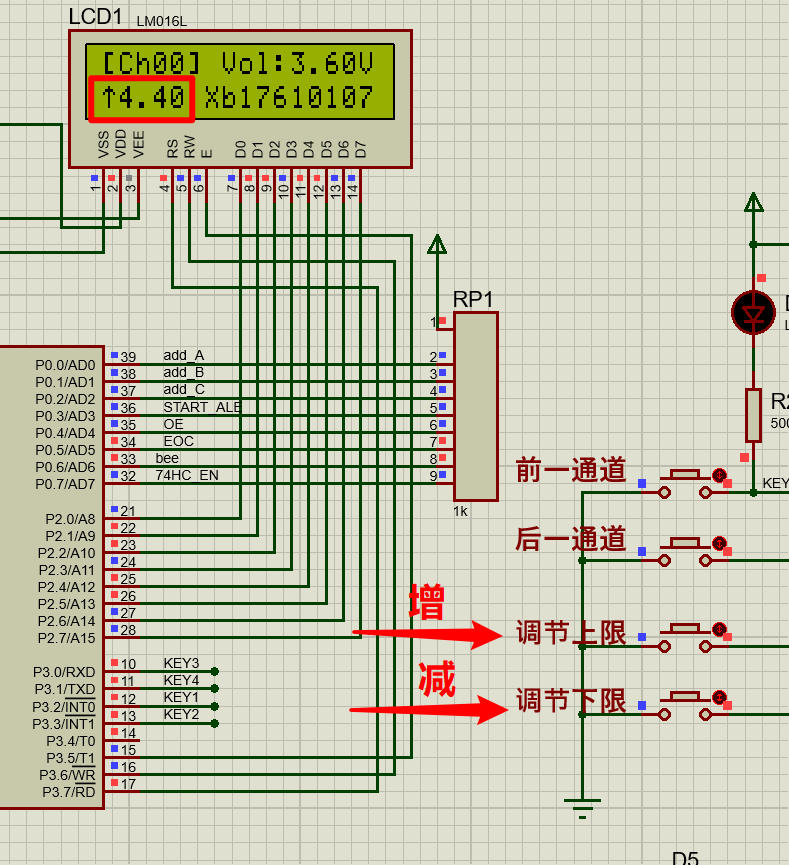
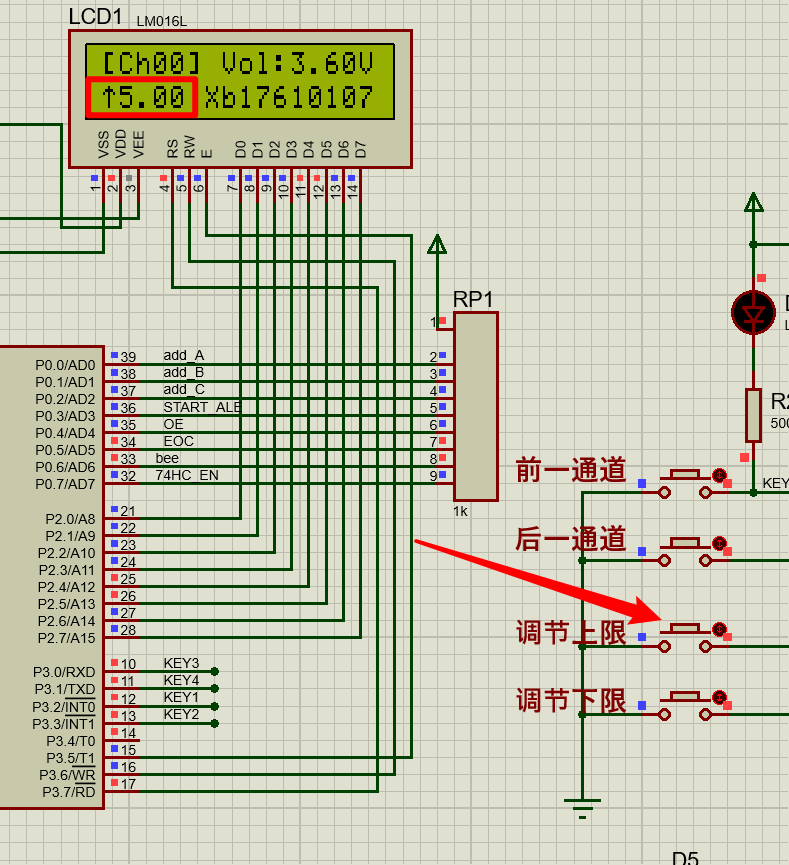


图 33-LED报警电路

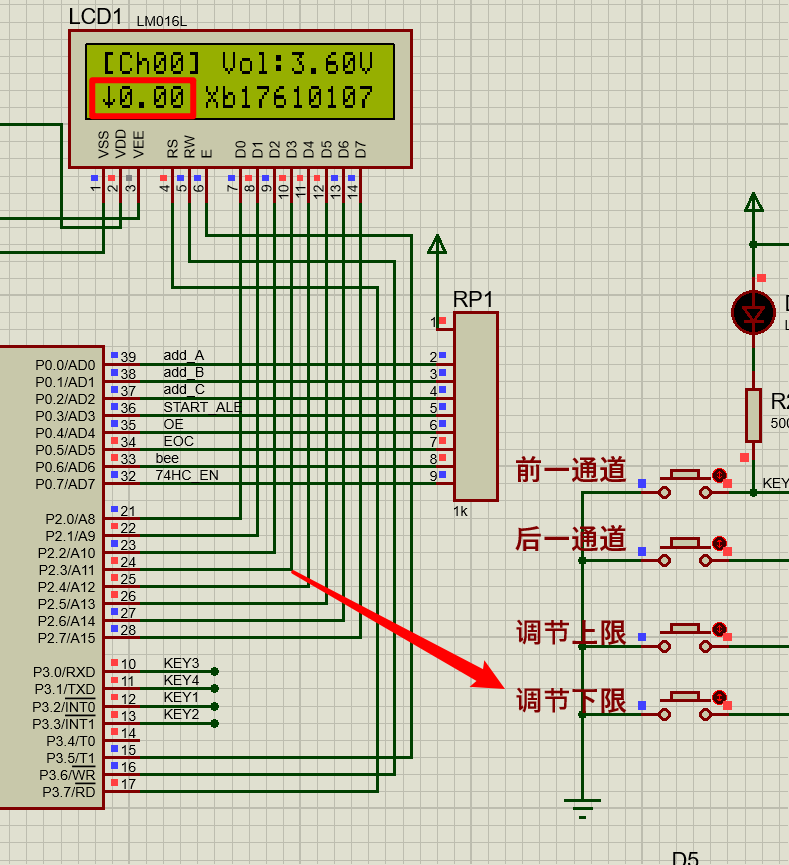
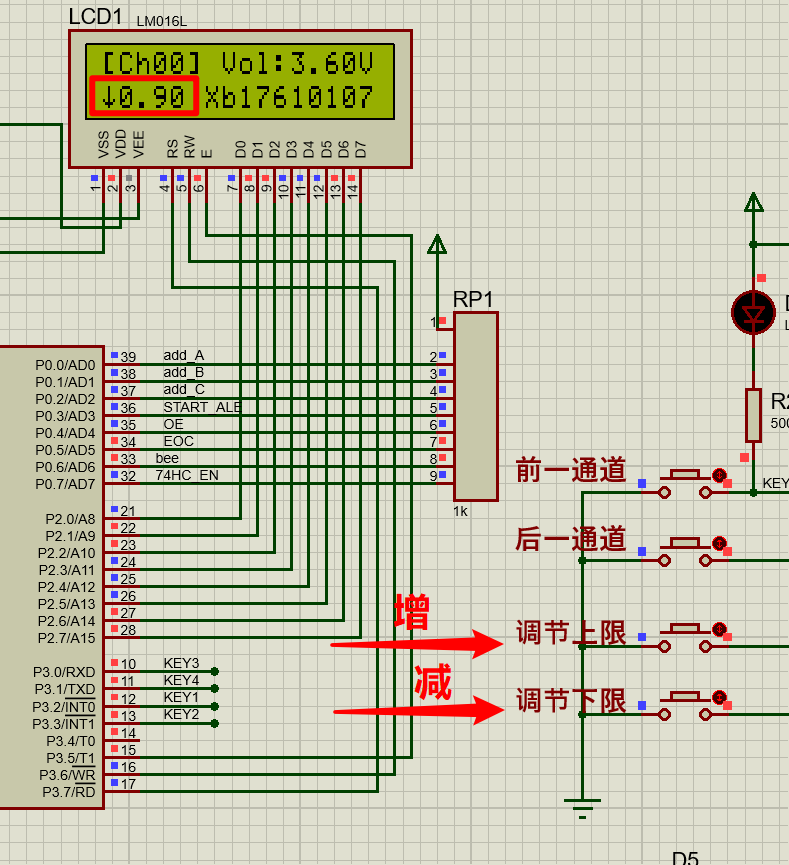
具体工作原理是，ABC为地址输入，Y0~Y7为译码输出，系统刚开始运行时，6脚受单片机控制E3=0，此时Y0~Y7输出为00000000，LED全灭。当处在某一通道下，并且该通道的电压异常，即超出上限或低于下限时，E3=1，译码输出，点亮该通道的报警LED实现与人的交互。

接下来时具体的调试现象，上下限电压由按键KEY3和KEY4控制，刚开始，第一次按下如果是KEY3，则进入上限电压调节界面，第二次按下任意KEY3或KEY4可以控制上限电压的增减。同理第一次按下如果是KEY4，则进入下限电压调节界面，第二次按下任意KEY3或KEY4可以控制下限电压的增减。

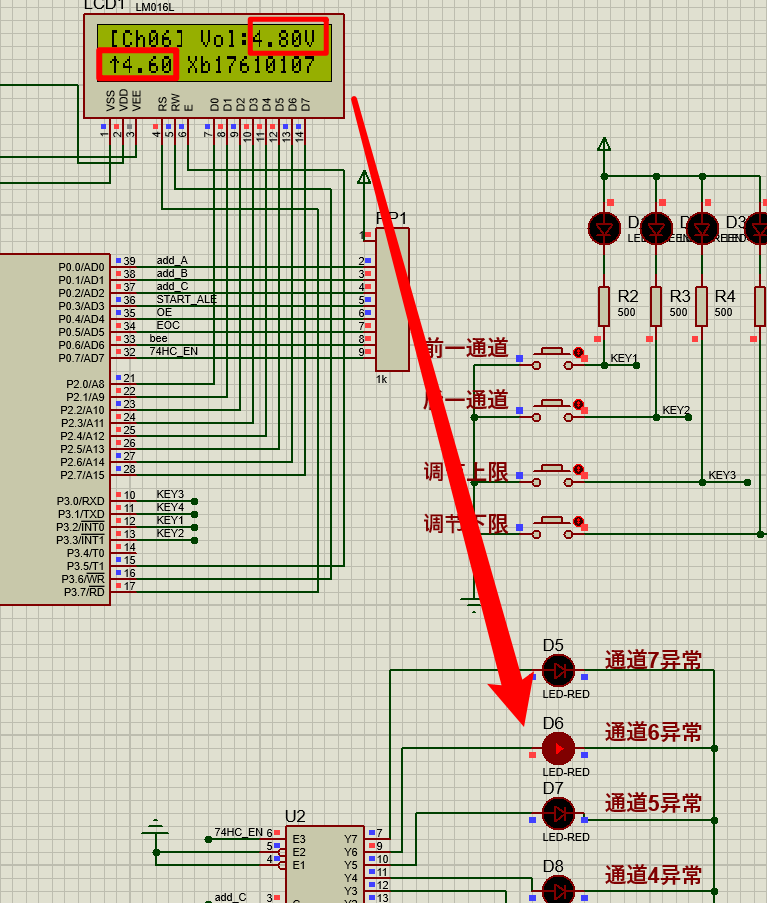
如果3秒没有操作，调节界面消失，但是该通道的上下限电压已更改。用户还需调节时，只需重复上述步骤即可。



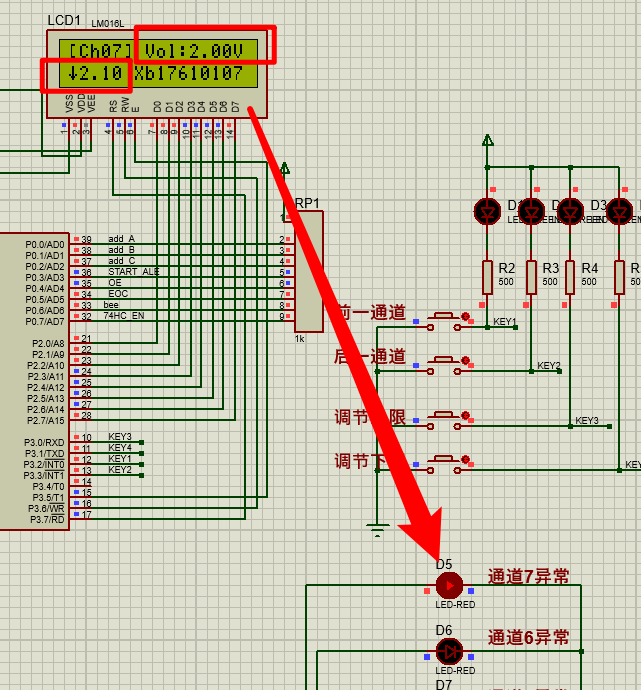
如上图所示，第一次按下如果是KEY3进入上限电压调节界面(最左边有上限电压箭头提示)，第二次按下任意KEY3或KEY4可以控制上限电压的增减。



如上图所示，第一次按下如果是KEY4进入下限电压调节界面(最左边有下限电压箭头提示)，第二次按下任意KEY3或KEY4可以控制下限电压的增减。



如上图所示，此时通道6的电压已经超过了用户调节的上限电压值，通道6的报警LED亮起。



如上图所示，此时通道7的电压已经低于了用户调节的下限电压值，通道7的报警LED亮起。其他通道的工作情况同上，此处不再进行赘述。

# 遇到的问题与解决方法

1. **按键模块按下输出0滞留**

标准的按键模块由上拉电阻构成，正常输出为1，按键按下时，0持续一瞬间，立马恢复高电平。但是仿真时，按键按下后低电平0会保持3秒，之后才恢复高电平1。将电压探头放置于按键端，发现会有惊人的充放电的效果。之后发现是由于仿真中加入了示波器检测波形，将示波器删去后恢复正常。将此问题归结为Proteus软件的问题。

1. **LCD电压显示始终为整数**

不管怎么改变输入电压的值，显示的电压始终是整数。发现在读取经ADC0809转化的数字量时，变量的类型没有考虑仔细，由于是整形int，参与运算必然丢位。准换成float，并且进行除法运算时注意除以浮点数51.0，解决问题。

# 心得体会

我始终认为电子信息工程是一个非常有潜力的专业，并且我对该专业充满了浓厚的兴趣。虽然说专业课让我学得喘不过气来，难度颇大，但我相信这只是一个阶段性的过渡期。

属于工科的电子信息工程需要我有缜密的分析思维，强大的动手能力和学习能力，这些素养还需后期慢慢培养。时光荏苒，如今我已是一名大三的人了，在大学度过的两年里，我收获了许许多多宝贵的经验，积累了一定的专业知识与素养，回首看看两年前的自己，我发现自己已不再是刚进大学的是那个对专业一无所知的小白了。我清楚地知道目前我学到的只是专业的冰山一角，剩下的还需要我跟随老师的脚步，课外自己慢慢摸索。

此次的多路数据采集控制系统运用到了很多之前专业课的知识，尤其是数字电子技术，模拟电子技术。落实到具体的课程设计我才恍然大悟专业知识并不是白学的，原来是这么运用的，也难怪很久以前专业课老师每次上课都会给我们埋下伏笔。以前自己还没有这么深的感触，但越到后来的专业课程设计我渐渐感觉到专业课知识起到对课程设计的支配与指导作用。

一连2次AT89C52的课程设计让我对单片机有了更加深入的理解。本次课程设计最突出的特点，也是最关键的地方是时序分析，严格按照时序图来编写程序。有ADC0809的时序图，还有LCD1602的时序图，之前一直看不懂这个时序图干些什么作用，现在可恍然大悟了。

2年下来，算算做过的课程实验加课程设计大概有30个左右，之前都是老师带着我们做着课程设计与实验。现在我逐渐摸清了要想做出一个能实现具体功能的成品需要哪几个步骤，常用哪几个模块，怎么把这几个模块联系起来。以我现在的能力，我能够自己设计出能够实现简易功能的成品，当然是建立在查阅资料的基础上。以我自己现在的知识储备量，我知道做出某种功能的成品，电路中运放（OP07逃不掉）是一定要用到的，运放电路常有电压比较器、差分比例放大电路（需要两个输入信号）、同向比例运算电路（其中一个特例可实现电压跟随器）；有运放就会涉及到正负供电，供电的话就需要电源模块，电源模块有两类，恒流源和恒压源。

我相信之后我还会遇到更大的困难，更大的挑战，不过本身也不就是一路解决问题，克服困难，提升自我的过程么。未来是未知的，我只有不断进步，不断去适应困苦的环境，才能超越自我，达到新的高度。

感谢一路上帮助过我的同学和老师。

附件

模块化程序代码

Proteus 8 Professional仿真布局图一张

**主函数main.c**

#include <reg52.h>

#include "AT89C52\_init.h"

#include "lcd\_init.h"

#include "lcd\_display.h"

#include "delay.h"

#include "adjust\_voltage.h"

#include "key\_check.h"

#include "led\_warning.h"

sbit KEY1=P3^2;

sbit KEY2=P3^3;

sbit bee=P0^6;

sbit led\_en=P0^7;

unsigned char flag=0;//flag用来保存通道序号

//void song();

void T0\_initial();

int main()

{

led\_en=0;

lcd\_init();

T0\_initial();

INT\_init();

while(1)

{

display\_mynumber();

KEY\_Check();//按键检测，集成上下限电压显示

display\_channel\_Voltage(flag);//显示通道和电压

}

}

void key1() interrupt 0//按键1显示前1通道，递减1

{

if(flag==0) flag=7;//到顶则从7开始

else flag-=1;

}

void timer0() interrupt 1

{

led\_warn();

TH0=0x3C;

TL0=0x80;//每次进入中断赋初值15536,50ms检测一次

}

void key2() interrupt 2//按键2显示后1通道，递增1

{

if(flag==7) flag=0;//到底则从0开始

else flag+=1;

}

void T0\_initial()

{

TMOD=0x01;//无需配置C/(T的反)，因为单片机默认上电是低电平

TH0=0x3C;

TL0=0x80;//每次进入中断赋初值15536,50ms检测一次

ET0=1;//允许T0中断

TR0=1;//启动T0计数器

EA=1;//打开总中断

}

/\* void song()

{

bee=1;

delay(500);

bee=0;

delay(50000);

} \*/

**AT89C52\_init.c**

#include <reg52.h>

void INT\_init()//使能AT89C52外部中断0和外部中断1

{

EA=0;//关闭总中断屏蔽外部中断

IT0=1;//下降沿中断

EX0=1;//开启分中断开关

IT1=1;//下降沿中断

EX1=1;//开启分中断开关

EA=1;//开启总中断

}

**AT89C52\_init.h**

#ifndef \_AT89C52\_INIT\_H\_

#define \_AT89C52\_INIT\_H\_

extern void INT\_init();

#endif

**delay.c**

void delay(unsigned char n)

{

unsigned char i,j;

for(i=n;i>0;i--)

for(j=110;j>0;j--);

}

**delay.h**

#ifndef \_DELAY\_H\_

#define \_DELAY\_H\_

extern void delay(unsigned char n);

#endif

**lcd\_init.c**

#include <reg52.h>

#include "lcd\_init.h"

#include "delay.h"

sbit LCD\_RS=P3^7;

sbit LCD\_RW=P3^6;

sbit LCD\_EN=P3^5;

sbit add\_A=P0^0;

sbit add\_B=P0^1;

sbit add\_C=P0^2;//ABC为3为地址输入线，译码出8位通道

sbit START\_ALE=P0^3;//地址锁存，锁存CBA地址

sbit OE=P0^4;//转换后数据输出使能

sbit EOC=P0^5;//是否转换完成，0正忙，1转换成功

void lcd\_init()

{

LCD\_Write\_Command(0x38);//功能设置指令6：8位数据，双行显示，5\*7字形

LCD\_Write\_Command(0x0c);//显示开关控制指令4：开启显示屏，关光标，光标不闪烁

LCD\_Write\_Command(0x06);//置输入模式指令3：数据读写后光标右移，画面不移动

LCD\_Write\_Command(0x01);//清屏指令1

}

void LCD\_Write\_Command(unsigned char com)//写命令函数

{

LCD\_EN=0;

LCD\_RS=0;//选择指令寄存器

LCD\_RW=0;//选择写

P2=com;//传命令

delay(5);

LCD\_EN=1;//将使能线置高

delay(5);//将高电平延时1段时间

LCD\_EN=0;//产生下降沿，让LCD执行命令

}

void LCD\_Write\_Value(unsigned char value)//写数据函数

{

LCD\_EN=0;

LCD\_RS=1;//选择数据寄存器

LCD\_RW=0;//选择写

P2=value;//传输显示数据

delay(5);

LCD\_EN=1;//将使能线置高

delay(5);//将高电平延时1段时间

LCD\_EN=0;//产生下降沿，让LCD执行命令

}

**lcd\_init.h**

#ifndef \_LCD\_INIT\_H\_

#define \_LCD\_INIT\_H\_

extern void lcd\_init();

extern void LCD\_Write\_Command(unsigned char com);

extern void LCD\_Write\_Value(unsigned char value);

#endif

**lcd\_display.c**

#include "lcd\_init.h"

#include "lcd\_display.h"

#include "ADC.h"

#include "delay.h"

#include "adjust\_voltage.h"

unsigned char selfchar\_up[]={0x04,0x0E,0x15,0x04,0x04,0x04,0x04,0x00};

unsigned char selfchar\_down[]={0x04,0x04,0x04,0x04,0x15,0x0E,0x04,0x00};

unsigned char selfchar\_clear[]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

void display\_mynumber()//显示学号

{

LCD\_Write\_Command(0x46+0x80);//学号从第2行第7个开始显示

LCD\_Write\_Value(0x58);//显示X

//delay(20);

LCD\_Write\_Value(0x62);//显示b

//delay(20);

LCD\_Write\_Value(0x31);//显示1

//delay(20);

LCD\_Write\_Value(0x37);//显示7

//delay(20);

LCD\_Write\_Value(0x36);//显示6

//delay(20);

LCD\_Write\_Value(0x31);//显示1

//delay(20);

LCD\_Write\_Value(0x30);//显示0

//delay(20);

LCD\_Write\_Value(0x31);//显示1

//delay(20);

LCD\_Write\_Value(0x30);//显示0

///delay(20);

LCD\_Write\_Value(0x37);//显示7

//delay(20);

}

void display\_channel\_Voltage(unsigned char m)//显示通道和电压

{

LCD\_Write\_Command(0x00+0x80);//送通道号显示初地址

LCD\_Write\_Value(0x5b);//显示[

LCD\_Write\_Value(0x43);//显示C

LCD\_Write\_Value(0x68);//显示h

LCD\_Write\_Value(0x30);//显示0

LCD\_Write\_Value(flag+0x30);//显示0

LCD\_Write\_Value(0x5d);//显示]

//LCD\_Write\_Value(0x3A);//显示:

LCD\_Write\_Command(0x07+0x80);//送电压显示初地址

LCD\_Write\_Value(0x56);//显示V

LCD\_Write\_Value(0x6F);//显示o

LCD\_Write\_Value(0x6c);//显示l

LCD\_Write\_Value(0x3A);//显示:

switch(m)

{

case 0:

ADC(0);

LCD\_Write\_Value(temp2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((temp2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(temp2%10+0x30);//显示第2位小数

break;

case 1:

ADC(1);

LCD\_Write\_Value(temp2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((temp2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(temp2%10+0x30);//显示第2位小数

break;

case 2:

ADC(2);

LCD\_Write\_Value(temp2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((temp2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(temp2%10+0x30);//显示第2位小数

break;

case 3:

ADC(3);

LCD\_Write\_Value(temp2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((temp2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(temp2%10+0x30);//显示第2位小数

break;

case 4:

ADC(4);

LCD\_Write\_Value(temp2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((temp2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(temp2%10+0x30);//显示第2位小数

break;

case 5:

ADC(5);

LCD\_Write\_Value(temp2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((temp2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(temp2%10+0x30);//显示第2位小数

break;

case 6:

ADC(6);

LCD\_Write\_Value(temp2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((temp2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(temp2%10+0x30);//显示第2位小数

break;

case 7:

ADC(7);

LCD\_Write\_Value(temp2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((temp2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(temp2%10+0x30);//显示第2位小数

break;

}

LCD\_Write\_Value(0x56);//显示V

}

void display\_down(unsigned char p)

{

switch(p)

{

case 0:

LCD\_Write\_Value(down0/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((down0/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(down0%10+0x30);//显示第2位小数

break;

case 1:

LCD\_Write\_Value(down1/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((down1/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(down1%10+0x30);//显示第2位小数

break;

case 2:

LCD\_Write\_Value(down2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((down2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(down2%10+0x30);//显示第2位小数

break;

case 3:

LCD\_Write\_Value(down3/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((down3/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(down3%10+0x30);//显示第2位小数

break;

case 4:

LCD\_Write\_Value(down4/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((down4/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(down4%10+0x30);//显示第2位小数

break;

case 5:

LCD\_Write\_Value(down5/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((down5/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(down5%10+0x30);//显示第2位小数

break;

case 6:

LCD\_Write\_Value(down6/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((down6/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(down6%10+0x30);//显示第2位小数

break;

case 7:

LCD\_Write\_Value(down7/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((down7/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(down7%10+0x30);//显示第2位小数

break;

}

}

void display\_up(unsigned char p)

{

switch(p)

{

case 0:

LCD\_Write\_Value(up0/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((up0/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(up0%10+0x30);//显示第2位小数

break;

case 1:

LCD\_Write\_Value(up1/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((up1/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(up1%10+0x30);//显示第2位小数

break;

case 2:

LCD\_Write\_Value(up2/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((up2/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(up2%10+0x30);//显示第2位小数

break;

case 3:

LCD\_Write\_Value(up3/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((up3/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(up3%10+0x30);//显示第2位小数

break;

case 4:

LCD\_Write\_Value(up4/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((up4/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(up4%10+0x30);//显示第2位小数

break;

case 5:

LCD\_Write\_Value(up5/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((up5/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(up5%10+0x30);//显示第2位小数

break;

case 6:

LCD\_Write\_Value(up6/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((up6/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(up6%10+0x30);//显示第2位小数

break;

case 7:

LCD\_Write\_Value(up7/100+0x30);//显示最高位

LCD\_Write\_Value(0x2E);//显示小数点

LCD\_Write\_Value((up7/10)%10+0x30);//显示第1位小数

LCD\_Write\_Value(up7%10+0x30);//显示第2位小数

break;

}

}

void display\_adjust(char s)//显示调节后上下限电压的内容，s用来选择显示上限电压还是下限电压

{

unsigned char m;

LCD\_Write\_Command(0x40);//设定CGRAM地址，把自定义字符存储进去

switch(s)

{

case -1:

for(m=0;m<8;m++) //将selfchar\_down[]中的数据依次写入CGRAM的8位里

{

LCD\_Write\_Value(selfchar\_down[m]);

delay(5);

}

LCD\_Write\_Command(0x40+0x80);

LCD\_Write\_Value(0x00);

LCD\_Write\_Command(0x41+0x80);//送下限电压显示初地址

display\_down(flag);//显示对应通道的下限电压

break;

case 1:

for(m=0;m<8;m++) //将selfchar\_up[]中的数据依次写入CGRAM的8位里

{

LCD\_Write\_Value(selfchar\_up[m]);

delay(5);

}

LCD\_Write\_Command(0x40+0x80);

LCD\_Write\_Value(0x00);

LCD\_Write\_Command(0x41+0x80);//送上限电压显示初地址

display\_up(flag);//显示对应通道的上限电压

break;

case 0:

for(m=0;m<8;m++) //将selfchar\_clear[]中的数据依次写入CGRAM的8位里

{

LCD\_Write\_Value(selfchar\_clear[m]);

delay(5);

}

LCD\_Write\_Command(0x40+0x80);

LCD\_Write\_Value(0x00);

LCD\_Write\_Command(0x41+0x80);//送上限电压显示初地址

LCD\_Write\_Value(0x20);//显示空

LCD\_Write\_Value(0x20);//显示空

LCD\_Write\_Value(0x20);//显示空

LCD\_Write\_Value(0x20);//显示空

break;

}

}

**lcd\_display.h**

#ifndef \_LCD\_DISPLAY\_H\_

#define \_LCD\_DISPLAY\_H\_

extern void display\_mynumber();//显示学号

extern void display\_channel\_Voltage(unsigned char m);//显示通道和电压

extern void display\_down(unsigned char p);//显示下限电压值

extern void display\_up(unsigned char p);//显示上限电压值

extern void display\_adjust(char s);//显示调节后上下限电压的内容，s用来选择显示上限电压还是下限电压

extern unsigned char flag;//flag用来保存通道序号

#endif

**ADC.c**

#include <reg52.h>

#include "ADC.h"

#include "delay.h"

sbit add\_A=P0^0;

sbit add\_B=P0^1;

sbit add\_C=P0^2;//ABC为3为地址输入线，译码出8位通道

sbit START\_ALE=P0^3;//地址锁存，锁存CBA地址

sbit OE=P0^4;//转换后数据输出使能

sbit EOC=P0^5;//是否转换完成，0正忙，1转换成功

float temp1=0;//temp1用来读取转换后的16进制

unsigned int temp2=0;//用来保存扩大啊100后的temp1

void ADC(unsigned char n)

{

char i;

switch(n)

{

case 0:

temp1=0;//由于0通道采取多次采样的方法，所以需要复位

for(i=5;i>0;i--)

{

START\_ALE=0;//

OE=0;

select\_channel(0);//选择通道0进行转化

START\_ALE=1;

delay(5);

START\_ALE=0;

delay(5);//等待AD转换完成

while(EOC==1)

{

P1=0xff;//读数据之前将P1口的电压置高

OE=1;

temp1=temp1+P1/51.0;//将转换后的16进制准换成电压

delay(5);

OE=0;

break;

}

}

temp2=(temp1/5.0)\*100;

break;

case 1:

START\_ALE=0;

OE=0;

select\_channel(1);//选择通道1进行转化

START\_ALE=1;

delay(5);

START\_ALE=0;

delay(5);//等待AD转换完成

while(EOC==1)

{

P1=0xff;//读数据之前将P1口的电压置高

OE=1;

temp1=P1/51.0;//将转换后的16进制准换成电压

temp2=temp1\*100;

delay(5);

OE=0;

break;

}

break;

case 2:

START\_ALE=0;

OE=0;

select\_channel(2);//选择通道2进行转化

START\_ALE=1;

delay(5);

START\_ALE=0;

delay(5);//等待AD转换完成

while(EOC==1)

{

P1=0xff;//读数据之前将P1口的电压置高

OE=1;

temp1=P1/51.0;//将转换后的16进制准换成电压

temp2=temp1\*100;

delay(5);

OE=0;

break;

}

break;

case 3:

START\_ALE=0;

OE=0;

select\_channel(3);//选择通道3进行转化

START\_ALE=1;

delay(5);

START\_ALE=0;

delay(5);//等待AD转换完成

while(EOC==1)

{

P1=0xff;//读数据之前将P1口的电压置高

OE=1;

temp1=P1/51.0;//将转换后的16进制准换成电压

temp2=temp1\*100;

delay(5);

OE=0;

break;

}

break;

case 4:

START\_ALE=0;

OE=0;

select\_channel(4);//选择通道4进行转化

START\_ALE=1;

delay(5);

START\_ALE=0;

delay(5);//等待AD转换完成

while(EOC==1)

{

P1=0xff;//读数据之前将P1口的电压置高

OE=1;

temp1=P1/51.0;//将转换后的16进制准换成电压

temp2=temp1\*100;

delay(5);

OE=0;

break;

}

break;

case 5:

START\_ALE=0;

OE=0;

select\_channel(5);//选择通道5进行转化

START\_ALE=1;

delay(5);

START\_ALE=0;

delay(5);//等待AD转换完成

while(EOC==1)

{

P1=0xff;//读数据之前将P1口的电压置高

OE=1;

temp1=P1/51.0;//将转换后的16进制准换成电压

temp2=temp1\*100;

delay(5);

OE=0;

break;

}

break;

case 6:

START\_ALE=0;

OE=0;

select\_channel(6);//选择通道6进行转化

START\_ALE=1;

delay(5);

START\_ALE=0;

delay(5);//等待AD转换完成

while(EOC==1)

{

P1=0xff;//读数据之前将P1口的电压置高

OE=1;

temp1=P1/51.0;//将转换后的16进制准换成电压

temp2=temp1\*100;

delay(5);

OE=0;

break;

}

break;

case 7:

START\_ALE=0;

OE=0;

select\_channel(7);//选择通道7进行转化

START\_ALE=1;

delay(5);

START\_ALE=0;

delay(5);//等待AD转换完成

while(EOC==1)

{

P1=0xff;//读数据之前将P1口的电压置高

OE=1;

temp1=P1/51.0;//将转换后的16进制准换成电压

temp2=temp1\*100;

delay(5);

OE=0;

break;

}

break;

}

}

void select\_channel(unsigned char n)//通道选择函数

{

switch(n)

{

case 0: add\_C=0; add\_B=0; add\_A=0; break;//开启通道0

case 1: add\_C=0; add\_B=0; add\_A=1; break;//开启通道1

case 2: add\_C=0; add\_B=1; add\_A=0; break;//开启通道2

case 3: add\_C=0; add\_B=1; add\_A=1; break;//开启通道3

case 4: add\_C=1; add\_B=0; add\_A=0; break;//开启通道4

case 5: add\_C=1; add\_B=0; add\_A=1; break;//开启通道5

case 6: add\_C=1; add\_B=1; add\_A=0; break;//开启通道6

case 7: add\_C=1; add\_B=1; add\_A=1; break;//开启通道7

}

}

**ADC.h**

#ifndef \_ADC\_H\_

#define \_ADC\_H\_

extern void ADC(unsigned char n);

extern void select\_channel(unsigned char n);//通道选择函数

extern unsigned int temp2;

#endif

**adjust\_voltage.c**

#include "adjust\_voltage.h"

unsigned int down0=0,up0=500;//通道0的上下限值

unsigned int down1=0,up1=500;//通道1的上下限值

unsigned int down2=0,up2=500;//通道2的上下限值

unsigned int down3=0,up3=500;//通道3的上下限值

unsigned int down4=0,up4=500;//通道4的上下限值

unsigned int down5=0,up5=500;//通道5的上下限值

unsigned int down6=0,up6=500;//通道6的上下限值

unsigned int down7=0,up7=500;//通道7的上下限值

void adjust\_up\_add(unsigned char p)//调节对应通道上限电压值向上

{

switch(p)

{

case 0:

if((up0+10)<=500) up0+=10;

break;

case 1:

if((up1+10)<=500) up1+=10;

break;

case 2:

if((up2+10)<=500) up2+=10;

break;

case 3:

if((up3+10)<=500) up3+=10;

break;

case 4:

if((up4+10)<=500) up4+=10;

break;

case 5:

if((up5+10)<=500) up5+=10;

break;

case 6:

if((up6+10)<=500) up6+=10;

break;

case 7:

if((up7+10)<=500) up7+=10;

break;

}

}

void adjust\_up\_reduce(unsigned char p)//调节对应通道上限电压值向下

{

switch(p)

{

case 0:

if(up0>down0+10) up0-=10;

break;

case 1:

if(up1>down1+10) up1-=10;

break;

case 2:

if(up2>down2+10) up2-=10;

break;

case 3:

if(up3>down3+10) up3-=10;

break;

case 4:

if(up4>down4+10) up4-=10;

break;

case 5:

if(up5>down5+10) up5-=10;

break;

case 6:

if(up6>down6+10) up6-=10;

break;

case 7:

if(up7>down7+10) up7-=10;

break;

}

}

void adjust\_down\_add(unsigned char p)//调节对应通道下限电压值向上

{

switch(p)

{

case 0:

if(down0<(up0-10)) down0+=10;

break;

case 1:

if(down1<(up1-10)) down1+=10;

break;

case 2:

if(down2<(up2-10)) down2+=10;

break;

case 3:

if(down3<(up3-10)) down3+=10;

break;

case 4:

if(down4<(up4-10)) down4+=10;

break;

case 5:

if(down5<(up5-10)) down5+=10;

break;

case 6:

if(down6<(up6-10)) down6+=10;

break;

case 7:

if(down7<(up7-10)) down7+=10;

break;

}

}

void adjust\_down\_reduce(unsigned char p)//调节对应通道下限电压值向下

{

switch(p)

{

case 0:

if((down0-10)>=0) down0-=10;

break;

case 1:

if((down1-10)>=0) down1-=10;

break;

case 2:

if((down2-10)>=0) down2-=10;

break;

case 3:

if((down3-10)>=0) down3-=10;

break;

case 4:

if((down4-10)>=0) down4-=10;

break;

case 5:

if((down5-10)>=0) down5-=10;

break;

case 6:

if((down6-10)>=0) down6-=10;

break;

case 7:

if((down7-10)>=0) down7-=10;

break;

}

}

**adjust\_voltage.h**

#ifndef \_ADJUST\_VOLTAGE\_H\_

#define \_ADJUST\_VOLTAGE\_H\_

extern void adjust\_up\_add(unsigned char p);//调节对应通道上限电压值向上

extern void adjust\_up\_reduce(unsigned char p);//调节对应通道上限电压值向下

extern void adjust\_down\_add(unsigned char p);//调节对应通道下限电压值向上

extern void adjust\_down\_reduce(unsigned char p);//调节对应通道下限电压值向下

externunsigned int down0,down1,down2,down3,down4,down5,down6,down7,up0,up1,up2,up3,up4,up5,up6,up7,temp2;

#endif

**key\_check.c**

#include <reg52.h>

#include "adjust\_voltage.h"

#include "lcd\_display.h"

#include "key\_check.h"

sbit KEY3=P3^0;

sbit KEY4=P3^1;

void KEY\_Check()

{

static unsigned char now=1,n=0,i=0,j=0;//n用来控制刚开始按下进入哪个界面，第二次按下是对固定界面电压调节,i的作用是若按键长时间没变化,隐藏调节界面

//j的作用是停留在哪个界面上持续控制

unsigned char past,key=0;//key来区别那个按键按下

past=now;

if(KEY3==0) {key=3;now=0;}

else if(KEY4==0) {key=4;now=0;}

else now=1;

if((past==1)&&(now==0))

{

i=0;

if(n==0)///第一次按两个键中的任意一个是判断进入那个界面

{

if(key==3) {display\_adjust(1); j=1;}//1进入上限电压调节界面

else if(key==4) {display\_adjust(-1); j=-1;}//-1进入下限电压调节界面

n+=1;

}

else//第二次以后开始调节电压

{

switch(j)

{

case 1://对上限电压进行调节

if(key==3)

{

adjust\_up\_add(flag);

display\_adjust(1);

}

else if(key==4)

{

adjust\_up\_reduce(flag);

display\_adjust(1);

}

break;

case -1://对下限电压进行调节

if(key==3)

{

adjust\_down\_add(flag);

display\_adjust(-1);

}

else if(key==4)

{

adjust\_down\_reduce(flag);

display\_adjust(-1);

}

break;

}

}

}

else if((past==1)&&(now==1))

{

i++;

if(i==30) {display\_adjust(0); n=0;}

}

}

**key\_check.h**

#ifndef \_KEY\_CHECK\_H\_

#define \_KEY\_CHECK\_H\_

extern void KEY\_Check();//检测调节电压上下限的按键

#endif

**led\_warning.c**

#include <reg52.h>

#include "led\_warning.h"

#include "adjust\_voltage.h"

sbit led\_en=P0^7;

void led\_warn()

{

switch(flag)

{

case 0:

if((temp2>up0) || (temp2<down0)) led\_en=1;

else led\_en=0;

break;

case 1:

if((temp2>up1) || (temp2<down1)) led\_en=1;

else led\_en=0;

break;

case 2:

if((temp2>up2) || (temp2<down2)) led\_en=1;

else led\_en=0;

break;

case 3:

if((temp2>up3) || (temp2<down3)) led\_en=1;

else led\_en=0;

break;

case 4:

if((temp2>up4) || (temp2<down4)) led\_en=1;

else led\_en=0;

break;

case 5:

if((temp2>up5) || (temp2<down5)) led\_en=1;

else led\_en=0;

break;

case 6:

if((temp2>up6) || (temp2<down6)) led\_en=1;

else led\_en=0;

break;

case 7:

if((temp2>up7) || (temp2<down7)) led\_en=1;

else led\_en=0;

break;

}

}

**led\_warning.h**

#ifndef \_LED\_WARNING\_H\_

#define \_LED\_WARNING\_H\_

extern void led\_warn();//led警告

extern unsigned char flag;//flag用来保存通道序号

#endif